

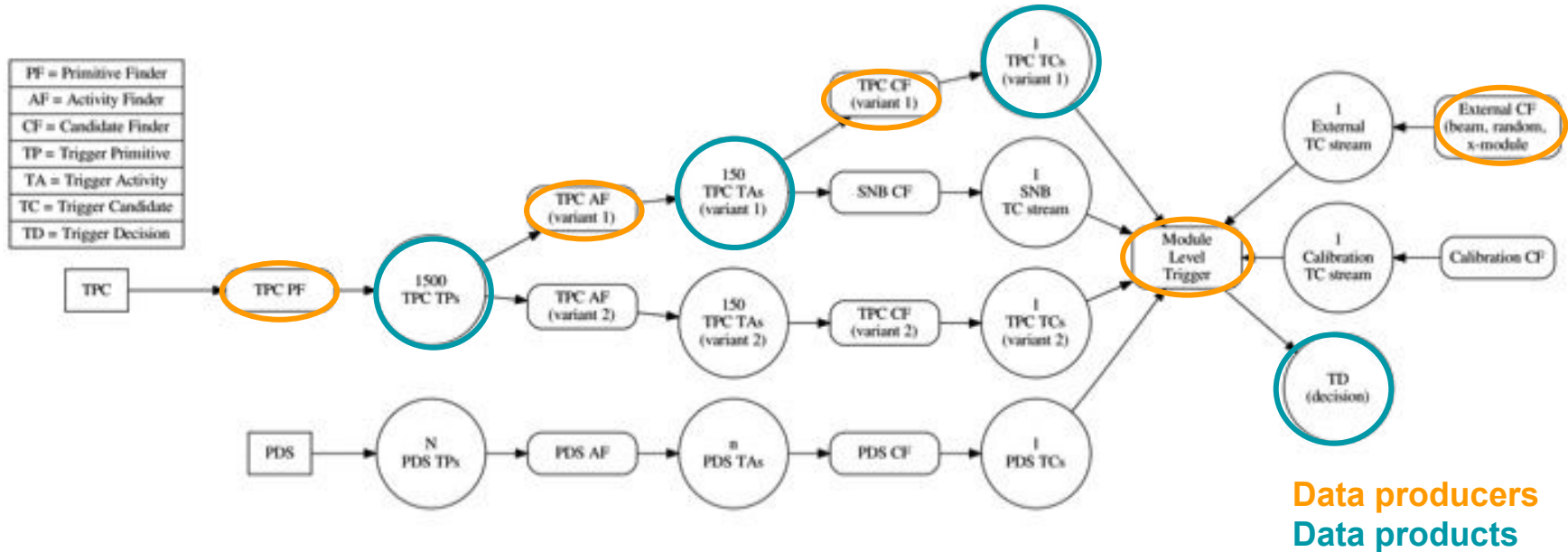
Finalizing DS data products

(Updated slides following feedback from Feb. 2 meeting)

Georgia Karagiorgi, Lukas Arnold
Columbia University

DUNE DAQ PP/DS WG Meeting
Feb. 9-19, 2021

Reminder: DS, operates ~independently per FD Module



See also dune [docdb-21839](#) for DS Specifications

*Not shown: HLF. The HLF provides its own data products (additional header info, and modified format data load). TBD after discussion with offline.
Care to be taken to avoid proliferation of formats.

TP Data: To be refined

The input of the TPC and PDS PFs is raw unbiased TPC data, and minimally biased PDS data, respectively, with format specified by the UD. The output of the TPC and PDS PF is Trigger Primitives (TPs), with TP format specified as follows:

```
struct TriggerPrimitive {  
    int64_t time_start = {0};  
    int64_t time_peak = {0};  
    int32_t time_over_threshold = {0};  
    uint32_t channel = {0};  
    uint32_t adc_integral = {0};  
    uint16_t adc_peak = {0};  
    uint32_t detid = {0};  
    uint32_t type = {0};  
    uint16_t algorithm = {0};  
    uint16_t version = {0};  
    uint32_t flag = {0};  
};
```

In units of 62.5 MHz time ticks. 62.5 MHz is the common system clock; anything is a timestamp in the DS is in these units.

These numbers are assigned in upstream DAQ.

If different units are used for TPC and PDS TPs, (e.g. 2 MHz and 62.5 MHz), then DS must know about relative conversion.

- Assuming all times are in 62.5 MHz clock, and assuming up to 8 hr runs, time_start and time_peak (which are absolute times) should be Up to ~42 bits long. If absolute clock time since ~1970, then 64 bits. **Leave at 64 bits.**
- Time_over_threshold is Delta_t, and unlikely to span longer than a drift, or a maximum readout window (5.4ms). Up to 19 bits long. **Leave at 32 bits.**

TP Data: To be refined

The input of the TPC and PDS PFs is raw unbiased TPC data, and minimally biased PDS data, respectively, with format specified by the UD. The output of the TPC and PDS PF is Trigger Primitives (TPs), with TP format specified as follows:

```
struct TriggerPrimitive {  
    int64_t time_start = {0};  
    int64_t time_peak = {0};  
    int32_t time_over_threshold = {0};  
uint16_t uint32_t channel = {0};  
    uint32_t adc_integral = {0};  
    uint16_t adc_peak = {0};  
uint16_t uint32_t detid = {0};  
uint16_t uint32_t type = {0};  
    uint16_t algorithm = {0};  
    uint16_t version = {0};  
    uint32_t flag = {0};  
};
```

Unique only within a subdetector ID, where subdetector ID is specified below.
16 bit would accommodate 65,536 unique channels per subdetector ID.

(Sub)detector ID, e.g. APA number and link number, or PDS group and link number, or subregion of VD charge readout, etc. Do we need to also specify module ID (1-4)?
Proposed format: 2 bits for module number, 8 bits for APA/PDS group/VD group number, 4 bits for link number. 16 bits total.

TPC VD, TPC HD, or PDS VD, or PDS HD. Any other types?

TP Data: To be refined

The input of the TPC and PDS PFs is raw unbiased TPC data, and minimally biased PDS data, respectively, with format specified by the UD. The output of the TPC and PDS PF is Trigger Primitives (TPs), with TP format specified as follows:

```
struct TriggerPrimitive {  
    int64_t time_start = {0};  
    int64_t time_peak = {0};  
    int32_t time_over_threshold = {0};  
uint16_t  
uint32_t channel = {0};  
    uint32_t adc_integral = {0};  
uint16_t  
uint32_t adc_peak = {0};  
uint16_t  
uint32_t detid = {0};  
uint16_t  
uint32_t type = {0};  
    uint16_t algorithm = {0};  
    uint16_t version = {0};  
    uint32_t flag = {0};  
};
```

Calculated in upstream DAQ. TP algorithm-specific.
12-bit ADC, so 16-bit ADC peak is okay. 5.4ms is 10,800 samples so
Maximum possible integral is $10,800 * 4096 \rightarrow$ 32-bit integral is sufficient.

The TP algorithm and version used in UD.
DS should be made aware of this through configuration.
UD should confirm format is okay.

Flags are provided from UD to indicate potential issues with TPs (e.g. UD buffer overflows or other errors). **How many bits to assign to this flag?**

TA Data: To be refined

One AF per PDS/TPC SubDetector unit (e.g. APA).

```
struct TriggerActivity {
```

```
int64_t time_start = {0};
```

```
int64_t time_end = {0};
```

```
int64_t time_peak = {0};
```

```
int64_t time_formation = {0};
```

In units of 62.5 MHz time ticks.

uint16_t

```
uint32_t channel_start = {0};
```

Quantify Activity extent in channel and time space. AF algorithm specific.

~~uint32_t~~

```
unit32_t channel_end = {0};
```

```
uint16_t
```

```
uint32_t channel_peak = {0};
```

```
uint64_t
```

```
uint32_t adc_integral = {0};
```

Quantify Activity properties. AF algorithm specific.

```
uint16_t adc_peak = {0};
```

uint16_t

```
uint32_t detid = {0};
```

Same as for TP Data.

```
uint32_t type = {0};
```

```
uint32_t algorithm = {0};
```

```
uint16_t version = {0};
```

```
std::vector<TriggerPrimitive> tp_list;
```

$$\};$$

A vector of TP's which make up this TA.

TA Data: To be refined

One AF per PDS/TPC SubDetector unit (e.g. APA).

```
struct TriggerActivity {  
    int64_t time_start = {0};  
    int64_t time_end = {0};  
    int64_t time_peak = {0};  
    int64_t time_formed = {0};  
    uint32_t channel_start = {0};  
    uint32_t channel_end = {0};  
    uint32_t channel_peak = {0};  
    uint64_t adc_integral = {0};  
    uint16_t adc_peak = {0};  
    uint32_t detid = {0};  
    uint32_t type = {0};  
    uint32_t algorithm = {0};  
    uint16_t version = {0};  
  
    std::vector<TriggerPrimitive> tp_list;  
};
```

activity

TA Type is recognizable by CF stage: e.g. TPC VD, TPC HD, PDS VD, PDS HD, "more aggressively masked/skipped noisy channels", ML, etc...

Both type and algorithm are assigned higher bits than for TP, due to higher-level/more diverse choices.

Specifies the TA algorithm and version used in DS

A vector of TP's which make up this TA.

Note: these will also exist in the trigger record. The latter will *also* contain *all other* TPs spanning the TD readout window (more inclusive than this list). (Another option is to provide TP ID's instead, and assign a TP ID as part of the TP Data.)

TC Data: To be refined

```
struct TriggerCandidate {  
    int64_t time_start = {0};  
    int64_t time_end = {0};  
    int64_t time_decided = {0};  
    uint32_t detid = {0};  
    uint32_t type = {0};  
    uint32_t algorithm = {0};  
    uint16_t version = {0};  
  
    std::vector<TriggerActivity> ta_list;  
};
```

Can be **per PDS/TPC SubDetector unit** (e.g. APA),
or **per group of PDS/TPC SubDetector units**.

A possible first implementation of CFs in DS is to
have two types of TPC CFs:

CF which works on APA-level

CF which works on module-level (all APAs)

TC Data: To be refined

Can be **per PDS/TPC SubDetector unit** (e.g. APA),
or **per group of PDS/TPC SubDetector units**.

```
struct TriggerCandidate {  
    int64_t time_start = {0};  
    int64_t time_end = {0};  
    int64_t time_decided candidate = {0};  
    uint32_t uint16_t detid = {0};  
    uint32_t type = {0};  
    uint32_t algorithm = {0};  
    uint16_t version = {0};  
  
    std::vector<TriggerActivity> ta_list;  
};
```

In units of 62.5 MHz time ticks.

Should be changed into a map of “2D channel-time components”: vector of (channel start, channel end, time start, time end) on per subdetector basis.

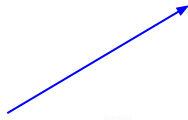
Fields for **channel_start**, **channel_end**, **channel_candidate** do not need to be added, because in principle they are retrievable from ta_list.

Expanded into a vector, to accommodate CFs which look at multiple (Sub)detector ID(s), e.g. APA number(s), or PDS group(s) associated with this TC.

TC Type is recognizable by MLT: e.g. TPC localized HE, TPC localized LE, TPC extended (SNB), PDS ..., External Trigger from TS, Calibration Trigger, SNEWS, ...

TD Data: To be refined

```
struct TriggerDecision {  
    int64_t time_start = {0};  
    int64_t time_end = {0};  
    int64_t time_trigger = {0};  
    uint32_t trigger_number = {0};  
    uint32_t run_number = {0};  
    uint32_t subrun_number = {0};  
    uint32_t type = {0};  
    uint32_t algorithm = {0};  
    uint16_t version = {0};  
std::map<ComponentIDs> td_components_list;  
  
    std::vector<TriggerCandidate> tc_list;  
};
```



Note:

There is redundancy in this data packet,
as this is what is to be stored as part of the trigger record.

The actual **TD message to the DFO** could/should be a
subset of this, and include only:

- **trigger_number**
- **trigger_type**
- **ComponentIDs** → a vector of <subdetector ID,
detid_time_start,detid_time_end>

Expanding Component IDs as above
allows DFO to read different time windows
at the APA-link granular level

where the 'ComponentIDs' map of includes a a list of detector IDs (e.g. APA number and APA link number, PDS number and PDS link number, etc.) to be read out. This list is to be informed by the detid's reported by the TAs forming a given TD.

Clarifications

1. The scheme presented allows the DFO to request data at the subdetector level. No further down-selection possible. (Until we get to High Level Filter, of course.)
2. `time_trigger` for data-driven triggers is ill-defined; it depends on the trigger type, as well as the extent of the event. For data-driven triggers, trigger time should be defined much more carefully and with higher resolution during reconstruction. On the other hand, for beam, or calibration, or random triggers, trigger time can and should be well defined.

For all data products: next steps

~~Is the correct number of bits ascribed to hold each piece of information?~~

Cross-check that the combined TP, TA, TC, TD rates are not overwhelming.
(Algorithm specific, but can be guesstimated).

Define a minimum list of TC types so that MLT development can proceed.

For each bundle of TPs, TAs, and TCs that get forwarded down the chain, a local timestamp must be added to the bundle so DS can keep track of its own trigger decision latency (relative to when TPs are received). This is especially needed for performance studies using emulated data.